



# node爬虫-从入门到进阶 副本

## 18 免责声明：

本文章涉及到的应用仅供学习交流使用，不得用于任何商业用途，数据来源于互联网公开内容，没有获取任何私有和有权限的信息（个人信息等）。由此引发的任何法律纠纷与本人无关！禁止将本文技术或者本文所关联的Github项目源码用于任何目的。

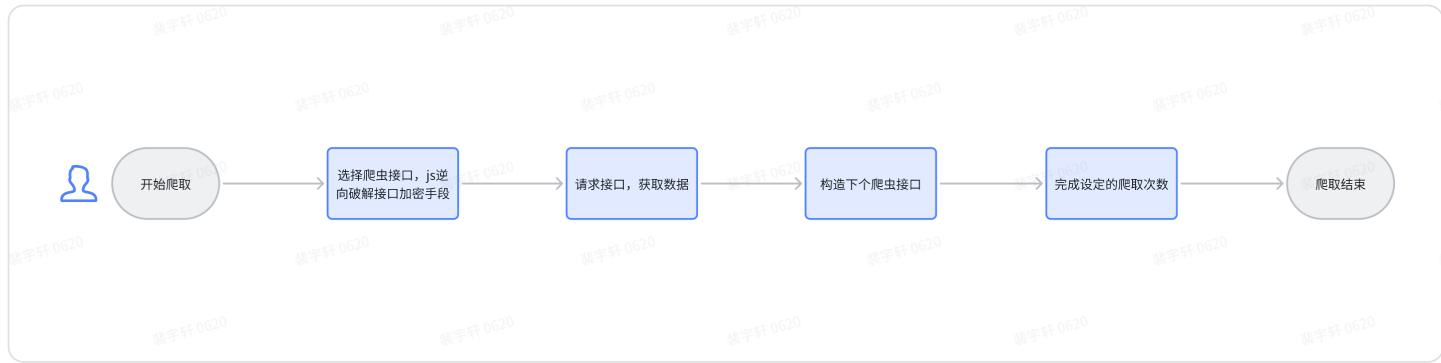
[抖音团队刑事法律培训——技术中立，爬虫无罪？](#)

## 一、创作背景

- 最近自己想尝试用gpt做下**续写小说、提炼小说大纲**等AIGC任务，想看看最后的效果，但是这些实验都需要大量的小说数据投喂，想要找到可以下载小说的网站或者app很不容易，而且下载的小说资源会出现**badcase**：最开始一些章节没有问题，但是后面的章节乱码等现象。于是探索了如何从各大官方网站上爬取相关章节（官网上的小说肯定没有错别字、乱码、缺少段落的情况），调研了node爬虫的姿势（不用python写，是因为大家都是前端工程师，对js会更加熟悉，以及发现puppeteer比python爬虫框架更加好用），写了一个通用的命令行小说爬虫项目。
- 从爬虫两种方式入手，让不太了解爬虫的同学，对爬虫的理解程度进阶；爬虫老司机也可以开阔思路，补充细节。也介绍反爬虫和反反爬虫的相关策略，让我们在开发过程中，对每个接口、模块、页面是否有爬虫风险了然于胸。

## 二、爬虫的方式

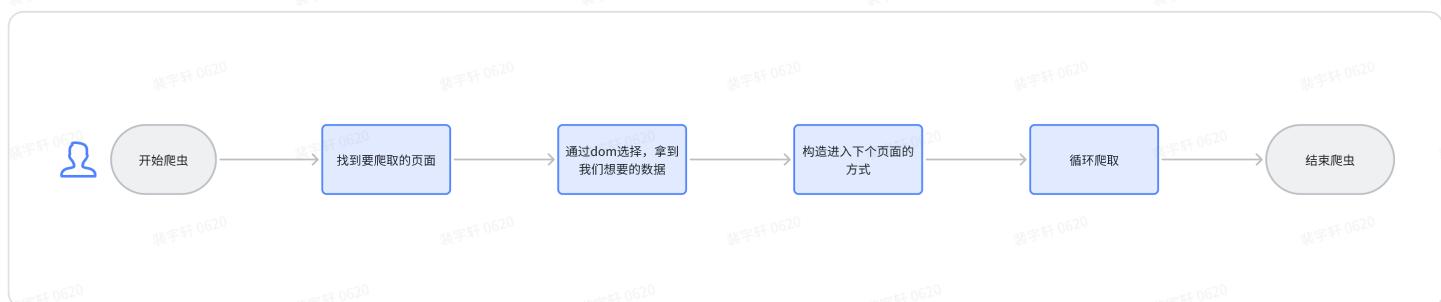
## 1. 接口请求式



爬取接口的原理始终都是一样的，那就是：

1. 设置好爬虫请求的相关参数（正常的浏览器User-Agent、需要携带的cookie等）
2. 通过浏览器devtools或者抓包工具筛选找到想要爬取的接口
3. 破解接口的安全加密手段
4. 构造下个接口请求的地址，找出规律
5. 开始爬虫
6. 将爬取的数据存在本地或者数据库中

## 2. 浏览器模拟请求式



当然我们也可以把思路扩大，放在整个浏览器展示页面中，直接获取页面展示的内容，而不是通过接口拿到数据了。

相比请求接口的爬虫方式，通过浏览器模拟有以下优点

1. 无需关注接口加密的方式，往往破解接口加密是爬虫工作中最困难的一步
2. 无需关注各种请求头的构造
3. 接口的反爬虫机制对浏览器访问基本失效，不用耗费精力在反爬虫机制上（比如接口请求qps、请求时间间隔等）

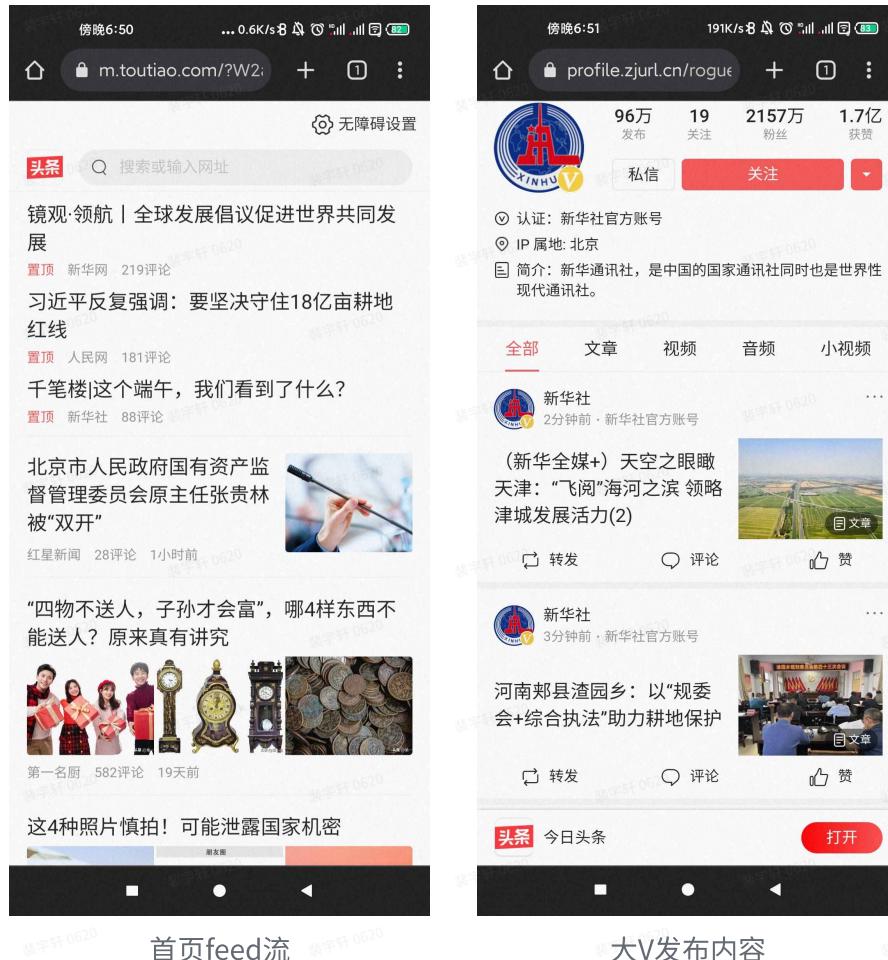
当然也会有一些缺点

1. 会请求很多不必要的资源，比如图片、js、css文件等等
2. 爬取速度较慢，因为模拟浏览器一个一个打开页面本就比请求接口慢

接下来对两种爬虫方式，使用真实的例子来进行解释

### 三、今日头条爬虫（接口爬虫方式）

请求接口的爬虫我们以爬取今日头条feed推送流以及各个大V的全部发表内容为案例



首页feed流

大V发布内容

### 流程图

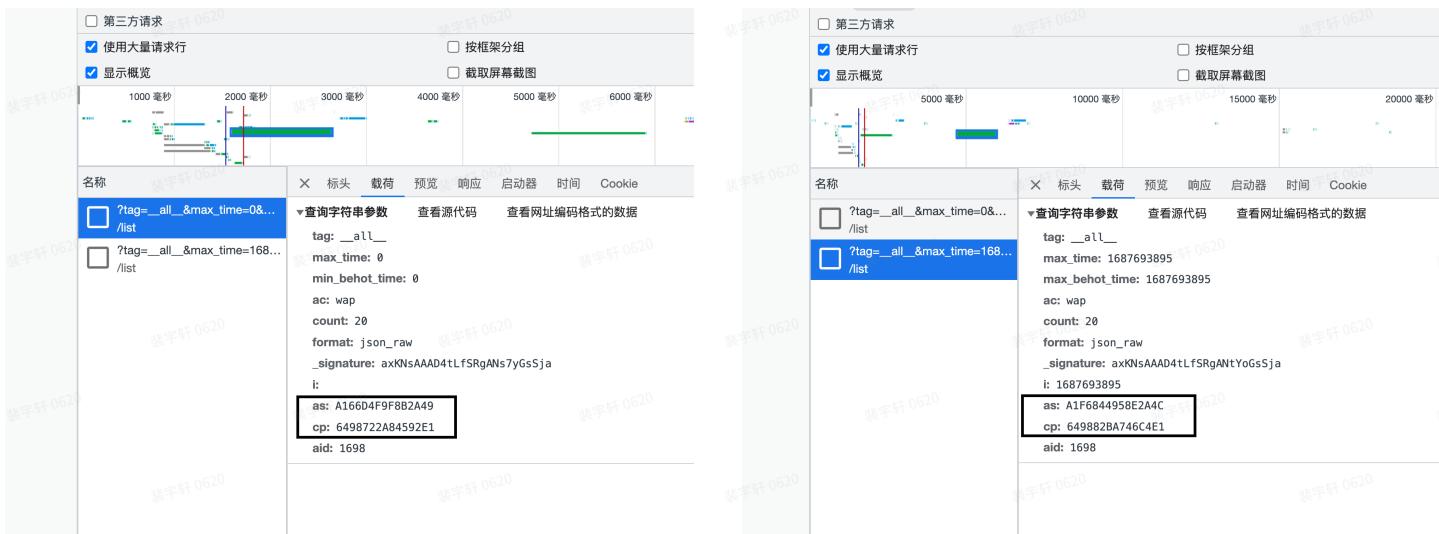


### 选择爬取接口、js逆向破解加密方法

#### feed流接口

头条的旧版feed流接口之前为`https://www.toutiao.com/api/pc/feed`，由于被爬虫的情况猖獗，现在已经改版了接口，改为了`https://www.toutiao.com/api/pc/list/feed`，之前的一套加密方案已经失效。但是经过筛选观察，发现h5版头条的仍然使用类似旧版的加密方案，有很大爬虫风险。

## h5头条feed流接口 <https://m.toutiao.com/list/>



观察发现：请求存在参数as、cp两个参数，as开头均为A1，cp结尾均为E1。于是可以猜想这可能是两个拼接的字符串，于是在页面源代码中搜索这两个字符串，发现的确是存在类似的代码的。

并且下一个请求的max\_time、max\_behot\_time、i三个参数都是从前一个接口拿到的。

```
function M() {
    var e = Math.floor((new Date).getTime() / 1e3)
    , t = e.toString(16).toUpperCase()
    , r = W(e).toString().toUpperCase();
    if (8 !== t.length)
        return {
            as: "479BB4B7254C150",
            cp: "7E0AC8874BB0985"
        };
    for (var n = r.slice(0, 5), i = r.slice(-5), o = "", a = 0; a < 5; a++)
        o += n[a] + t[a];
    for (var s = "", l = 0; l < 5; l++)
        s += t[l + 3] + i[l];
    return {
        as: "A1".concat(o).concat(t.slice(-3)),
        cp: "".concat(t.slice(0, 3) + s, "E1")
    }
}
var v = r(9669)
, o = r.n(V)().create({
    timeout: 5e3
});
function B(e) {
    var t = (e || {}).max_time;
    return O.get("/list/", {
        params: (0,
        C.Z)((0,
        C.Z)({}), e), {}, {
            ac: "wap",
            ...
        }
    })
}
```

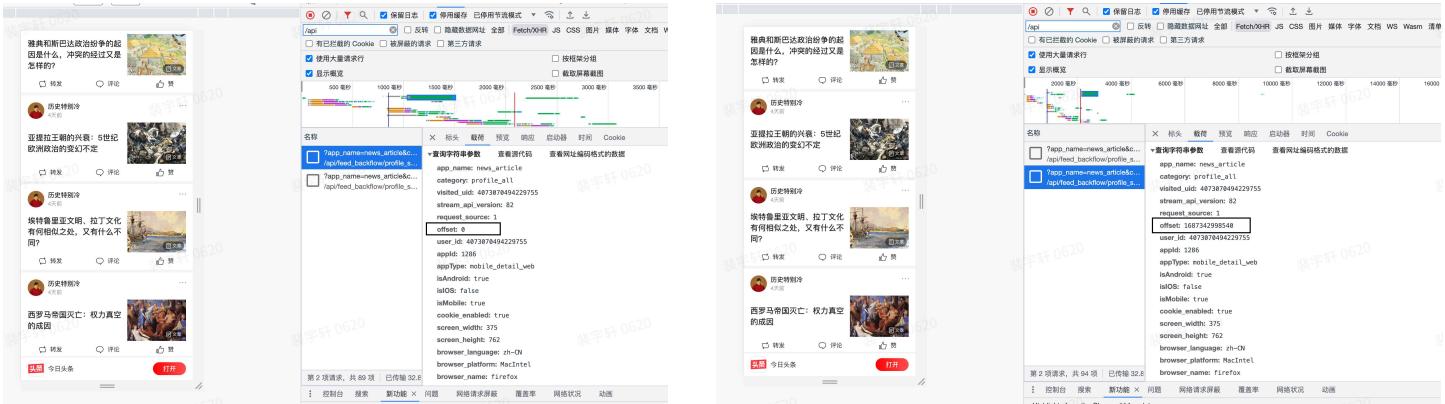
这便是加密函数的具体实现，我们可以猜想下，这里W()函数可能是一种hash摘要或者加密算法。可以尝试md5、SHA系列函数尝试，最后发现这里就是md5摘要。于是我们得到了接口加密的方式，如下所示：

```
1 const getAsCp = () => {
2     let result = {};
3     let now = Math.floor(Date.now() / 1000);
```

```
4 // let now = 1686993969;
5 console.log("now:", now);
6 let now_16 = now.toString(16).toUpperCase();
7 console.log("now_16:", now_16);
8
9 let md5 = CryptoJS.MD5(String(now)).toString().toUpperCase();
10
11 console.log("md5:", md5);
12
13 if (now_16.length !== 8) {
14     result = { as: "479BB4B7254C150", cp: "7E0AC8874BB0985" };
15     return result;
16 }
17
18 let first_string = md5.slice(0, 5);
19 let second_string = md5.slice(-5);
20 let first_result_string = "";
21 let second_result_string = "";
22
23 for (let i = 0; i < 5; i++) {
24     first_result_string += now_16[i + 3] + second_string[i];
25     second_result_string += first_string[i] + now_16[i];
26 }
27
28 result = {
29     as: "A1" + second_result_string + now_16.slice(-3),
30     cp: now_16.slice(0, 3) + first_result_string + "E1",
31 };
32
33 console.log("ascp:", result);
34 return result;
35 };
```

## 用户页面接口

用户页数据接口: [https://profile.zjurl.cn/api/feed\\_backflow/profile\\_share/v1/](https://profile.zjurl.cn/api/feed_backflow/profile_share/v1/)



观察发现仅仅offset偏移量存在着变化，并无加密参数。但是我们要获取到每个大V的uid  
下一个请求的offset参数同样是从前一个接口拿到的。

## 携带配置项，开始爬取

给请求构造header和cookie（应对接口反爬虫策略），发起请求。

```

1 const headers = {
2   "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36",
3   // 省略. . . .
4   "Sec-Fetch-User": "?1",
5   "Upgrade-Insecure-Requests": "1",
6 };
7 }
8
9 const cookies = {
10   tt_webid: "7072661175557015053",
11 };
12
13 const getData = async (url, headers, cookies) => {
14   const resp = await axios.get(url, {
15     headers,
16     cookie: cookies,
17   },
18   );
19 };
20
21 console.log("url:", url);
22
23 const data = resp.data;
24 return data;
25 };

```

## 构造下个接口，循环爬取

请求过程中发现，即使携带了header请求头以及cookie，仍然会被接口层的反扒策略发现，会一直返回相同的数据，也就是fake假数据。

```
标题: 【妖刀姬】这么大的你能把握住吗? zbrush妖刀姬模型雕刻教学
新闻链接: https://toutiao.com/i7248253232814752289/
头条号: 3dmax官方
评论数: 4
31
标题: 打破历史气温纪录! 今年夏天有多热? 高温将持续多久?
新闻链接: https://toutiao.com/i7248511307790418470/
头条号: 打量视频
评论数: 9
31
标题: 美媒披露: 国务院要求各使馆保持缄默
新闻链接: https://toutiao.com/i7248418098401853987/
头条号: 环球网
评论数: 177
31
标题: 二战电影, 铁十字勋章1
新闻链接: https://toutiao.com/i7248436412633055784/
头条号: 自在光头胜
评论数: 0
31
标题: 一口气看完《心理罪》
新闻链接: https://toutiao.com/i7248295941726274087/
头条号: 魔法钢琴gRD
评论数: 21
31
标题: “德艺双馨”书画家涉嫌重大刑案被悬赏? 多个协会发声: 他不是我们会员
新闻链接: https://toutiao.com/i7248592826073629242/
头条号: 极目新闻
评论数: 1737
31
标题: 普京: 将采取果断行动稳定局势
新闻链接: https://toutiao.com/item/7248260756733477391/
头条号: 光明网
评论数: 2910
31
标题: 20230521《看东方》
新闻链接: https://toutiao.com/i7235425919328846376/
头条号: 看看新闻
评论数: 1308
31
now: 1687696855
now_16: 649835D7
md5: C0177054BC4CB45D9AB10EB085FC190D
ascp: { as: 'A1C6041978735D7', cp: '6498C3159D07DE1' }
url: https://m.toutiao.com/list/?tag=__all__&max_time=1687665284&max_behot_time=1687665284&as=A1C6041978735D7&cp=6498C3159D07DE1&son_raw&aid=1698&_signature=
```

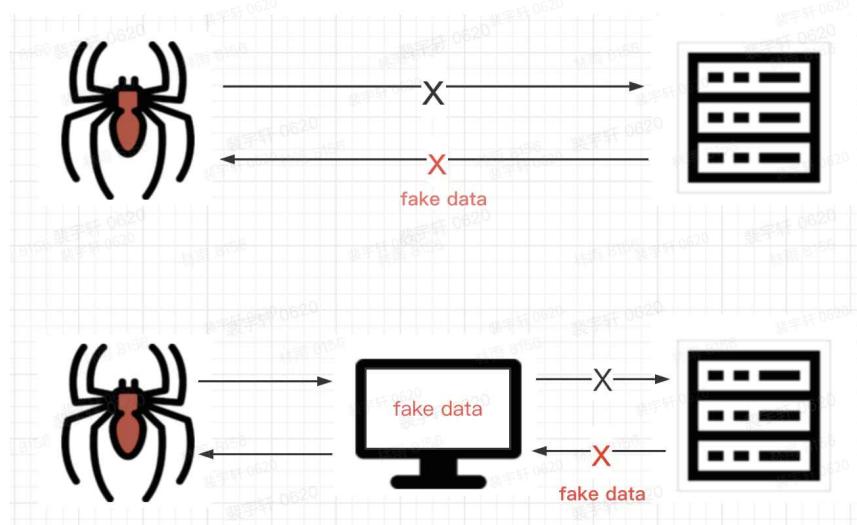
爬取的数据会一直重复

这是头条的蜜罐反爬虫手段，可以参考下面的文档。

[头条PC个人主页反爬方案](#)

## 蜜罐方案

思想：当服务端识别出爬虫后，接口返回随机的假数值，误导爬虫。



优点：只依赖于服务端，前端不参与。

缺点：比较依赖于爬虫识别的准确率，可能出现正常用户看到fake data。

纵使携带了大量真实请求的请求头和cookie，仍然能被接口反爬虫给识别出来，那该怎么办呢？

于是我们的“大杀器”——无头浏览器就出场了。

可以在puppeteer中发起请求，现在我们本身就是在浏览器中发起的请求，自然各种浏览器参数都携带了，反爬虫手段就很难识别出来了。下面是用puppeteer爬取的数据

新闻链接: <https://toutiao.com/i7248532172070535740/>  
 头条号: 启农说  
 评论数: 0

**121**  
 标题: 2000名女学生无一考入清华北大,就该被批评?张桂梅的回应显格局  
 新闻链接: <https://toutiao.com/i7248535912035156518/>  
 头条号: 古今探  
 评论数: 3

**121**  
 标题: 中方发出警告后,两个北约国撞枪口上了,这一次中国不再客气!  
 新闻链接: <https://toutiao.com/i7248545041881612832/>

头条号: 橘子观察室  
 评论数: 40

**121**  
 标题: 一觉醒来,局势突然反转,俄军出动“末日飞机”,瓦格纳决定撤军  
 新闻链接: <https://toutiao.com/i7248471815600144896/>  
 头条号: 影史浮沉  
 评论数: 0

**121**  
 标题: 纪委通报,北京又有两位美女被查,严查大战正在进行  
 新闻链接: <https://toutiao.com/i7248419826081858081/>  
 头条号: 微笑鲸鱼ih0  
 评论数: 0

**121**  
 标题: 二战电影,铁十字勋章1  
 新闻链接: <https://toutiao.com/i7248436412633055784/>  
 头条号: 自在光头胜  
 评论数: 0

**121**  
 标题: 3小时空谷幽兰+舒缓轻音乐#乐女左符霜不袄,一尊时度少得归  
 新闻链接: <https://toutiao.com/i7220413245717217849/>  
 头条号: 茵簫羌音乐  
 评论数: 1

**121**  
 标题: 实拍中国新疆喀什古城,满街都是漂亮的维吾尔族女孩  
 新闻链接: <https://toutiao.com/i7164754479772860968/>  
 头条号: 七年环球  
 评论数: 36

**121**  
 now: 1687696984  
 now\_16: 64983658  
 md5: B7FB9701A480A1FCEC698DA756865E40  
 ascp: { as: 'A1B674F9B893658', cp: '64986356E5480E1' }  
 ↵  
 [2]+ Stopped node toutiao.js

可以一直爬下去

拿到足够多的feed流数据后,可以按照点赞数排序uid,为后续爬取大V账号做准备。

```

39 [
    {
        like_count: 28826, uid: '343500293151976' },
        like_count: 19689, uid: '3175006836628238' },
        like_count: 17689, uid: '1490553515288488' },
        like_count: 17191, uid: '312735617256476' },
        like_count: 14183, uid: '1161518517332376' },
        like_count: 10256, uid: '6731718723' },
        like_count: 7357, uid: '99023762470' },
        like_count: 7140, uid: '488636620217003' },
        like_count: 6932, uid: '3647305700' },
        like_count: 5181, uid: '6731718723' },
        like_count: 4309, uid: '61645190178' },
        like_count: 3916, uid: '6731718723' },
        like_count: 1805, uid: '839619068573352' },
        like_count: 1713, uid: '4384298192' },
        like_count: 1482, uid: '52308248531' },
        like_count: 1363, uid: '321521162160600' },
        like_count: 1288, uid: '52308248531' },
        like_count: 1000, uid: '452653888581837' },
        like_count: 935, uid: '980367136266163' },
        like_count: 289, uid: '5757425042' },
        like_count: 260, uid: '5954781019' },
        like_count: 201, uid: '1058170246015587' },
        like_count: 173, uid: '52029171784' },
        like_count: 130, uid: '2500718765030039' },
        like_count: 128, uid: '2758038604484333' },
        like_count: 69, uid: '57778602488' },
        like_count: 64, uid: '104235760134' },
        like_count: 45, uid: '106628403388' },
        like_count: 28, uid: '' },
        like_count: 7, uid: '2106279664302494' },
        like_count: 6, uid: '2093072693075629' },
        like_count: 6, uid: '734951374326814' },
        like_count: 5, uid: '4595880688' },
        like_count: 2, uid: '321521162160600' },
        like_count: 2, uid: '299530355227559' },
        like_count: 2, uid: '2940520598084413' },
        like_count: 1, uid: '2700834010576320' },
        like_count: 0, uid: '' },
        like_count: 0, uid: '4595880688' }
    ]
]

```

接下来就可以选择前几个uid来爬取他们的个人首页了。

```

请输入大主页爬取次数 (10) : 35
大V链接: https://profile.zjurl.cn/api/feed_backflow/profile_share/v1/?app_name=news_article&category=profile_all&stream_api_version=82&request_source=1&appId=1286&appType=mobile_detail_web&isAndroid=true&isIOS=false&isMobile=true&cookie_enabled=true&screen_width=375&screen_height=762&browser_language=zh-CN&browser_platform=MacIntel&browser_name=firefox&browser_version=114.0.0.0&browser_online=true&timezone_name=Asia%2FShanghai&visited_uid=343500293151976&user_id=343500293151976&offset=0&signature=
offset: 0
标题: 湖北一学霸放弃保送清华，说要享受高考过程，后来他考了多少分？
标题: 出卖米洛舍维奇以换取美国支持，他卖国求荣，结局比米洛舍维奇还惨
标题: 俄高官斥责中国低价购买“瓦良格号”，俄罗斯有权收回辽宁舰
标题: 女子耗时两年做试管手术喜得双胞胎，以为双喜临门，谁知是悲剧
标题: 李铁落网细节被曝光，曾有内部人士通知其出国，他却充满自信！
标题: 为讨好美国，韩国总统拿台湾问题祭纳投名状，韩外交部斥责中国无礼
标题: 2016年男子偷药店晾晒中药，泡酒喝下后身亡，家属将店主告上法庭
标题: 俄已无回头路，中国需重新考虑对俄关系？澳洲要与中国“一换”
标题: “李咏的妻子”哈文：丈夫病逝后坚持不改嫁，还定期给公婆生活费
标题: 父母近亲结婚，远嫁美国依旧命运坎坷的乌汶叻公主，一生特立独行
标题: 浙江富婆郑陈梅：爱上穷小子，送房送厂送宝马，却被其拳击身亡
标题: 从拒绝售卖到愿意售卖，俄罗斯对中国空军求购轰炸机的态度变化史
大V链接: https://profile.zjurl.cn/api/feed_backflow/profile_share/v1/?app_name=news_article&category=profile_all&stream_api_version=82&request_source=1&appId=1286&appType=mobile_detail_web&isAndroid=true&isIOS=false&isMobile=true&cookie_enabled=true&screen_width=375&screen_height=762&browser_language=zh-CN&browser_platform=MacIntel&browser_name=firefox&browser_version=114.0.0.0&browser_online=true&timezone_name=Asia%2FShanghai&visited_uid=343500293151976&user_id=343500293151976&offset=1687172341902&signature=
offset: 1687172341902
标题: 清代秦腔传播有什么内容？
标题: 先唐昆山文学有什么内容？如何发展？
标题: 高句丽对朝鲜李朝文人因何产生影响？
标题: 春秋文的表现方式有哪些，是否受地域影响？
标题: 韩国罗末丽初汉诗与中国文学的关系？如何影响？
标题: 汉代城乡关系的发展与内容？
标题: 汉唐间，巴蜀地区如何开发？有什么内容？
标题: 通过遥感调查，历史时期黑河下游的人类活动遗迹有哪些？
标题: 以《述善集》为中心，元代濮阳的西夏遗民对乡村建设有何优势？
标题: 明清藩封体制视角下的朝鲜王朝国家机构如何发展？
标题: 北宋宋神宗朝时期，政府是如何对西北地区进行战略决策的？
标题: 皇权与史权的合辙与分途—《明实录》中科举的历史与权力
大V链接: https://profile.zjurl.cn/api/feed_backflow/profile_share/v1/?app_name=news_article&category=profile_all&stream_api_version=82&request_source=1&appId=1286&appType=mobile_detail_web&isAndroid=true&isIOS=false&isMobile=true&cookie_enabled=true&screen_width=375&screen_height=762&browser_language=zh-CN&browser_platform=MacIntel&browser_name=firefox&browser_version=114.0.0.0&browser_online=true&timezone_name=Asia%2FShanghai&visited_uid=343500293151976&user_id=343500293151976&offset=1684925259847&signature=
offset: 1684925259847

```

至此，基于爬取接口方式的爬虫就介绍完毕了。拿到了这些账号uid、以及对应的作品数据，完全可以将其存在数据库中，用这些数据可以搭建自己的新闻网站（对应的正文、图片链接都能拿到）。

下面是头条爬取的示例

```
Code - 资源管理器 - toutiao.js M - toutiao.js - novel-crawler
toutiao.js > (e) crawlUser
329 await page.goto(url);
330
331 const res = await page.evaluate(async () => {
332   return JSON.parse(document.querySelector("pre").innerText);
333 })
334

问题 18 次控制台 JUPITER 快捷 GITLENS 输出
C02GFOLM6DV:novel-crawler bytedances
master+ 0 0 0 0 15
You, 1秒钟前 行 343, 列 48 空格:2 UTF-8 LF { JavaScript } 18 Spell Prettier JS
```

下面是头条做的一些反爬手段的文档，有兴趣的可以看看

## ④ 前端反爬虫方案讨论(数值、时间等场景)

## ④ 头条PC个人主页反爬方案

## 四、小说爬虫（模拟浏览器方式）

接下来到我们的“重头戏”了，来爬取各大文学网站的小说。

### 前置工作

为了让小说爬虫更具有通用性，需要给其添加通用配置项。等到爬取对应网站时，直接读取当前网站的配置即可。而写入配置最好是用json格式，但是传统json格式不能写注释，而为了后续阅读代码以及迭代是需要一定的注释的，于是配置项使用了hjson来定义。



主要作用就是在json文件中编写注释，地址：<https://github.com/hjson/hjson>

命令行工具则是用到了readline-sync这个包，用来获取命令行的输入，并在代码中使用输入。

A screenshot of the readlineSync npm package page on bnpm. The page shows basic information like version (v1.4.10), dependencies (none), and license (MIT). It includes a code example demonstrating how to interactively run a script via a console. To the right, there's a detailed package page with sections for Argus Security Scan (no report available), Weekly Downloads (13,675), Total Downloads (824,580), Maintainers (anseki), Version (1.4.10), Source (Public), Unpacked Size (132.87 kB), Total Files (9), Size (132.87 kB), and Last Publish (1 year ago).

地址：<https://bnpm.bytedance.net/package/readline-sync>

## 流程图



## 命令行输入配置

输入要爬取的小说地址、开始章节、结束章节、本地存储文件夹、是否合并章节、书源配置等信息，这些信息会在爬虫代码中获取到并使用。

```
C02GF07LMD6V:novel-crawler bytedance$ node index.js
>> cookie.json 不存在, 跳过设置 cookie...
Novel-Crawler 爬虫 v1.0.0
请输入开始章节网址: https://www.qidian.com/chapter/1031581171/685178735/
请输入起始章节 (1) : 1
请输入结束章节 (2000) : 800
请输入小说存放资料夹(预设: ./output)/书名:
请选择是否合并所有章节至单独文件?y/N(N) [y/n]: y
是否合并章节 true
请指定书源文件, 不指定则根据url自动匹配:
书源文件: ./bookSource/
获取书籍信息, 启动浏览器...
未指定书源文件, 根据url自动匹配...
>> 获取书源...
>> 书源文件: bookSource/qidian-com.hjson
>> 书源已确定: "qd小说", "https://www.qidian.com"
>> 书源初始化完成

启动浏览器, 获取书籍信息...
--- 正在获取小说信息 ---
书籍首页: https://book.qidian.com/info/1031581171/#Catalog
```

## 读取书源配置

一本小说的书籍信息包括了

- 书名
- 书籍图片
- 作者
- 书籍介绍
- 章节内容

要获取这些信息需要观察页面的dom构造，使用dom选择器拿到我对应的dom节点，再将节点文本信息去除掉多余的符号（换行符、空格等）。

在起点中文网书籍首页拿到以上信息，将这些配置写json里



```
// 从小说目录页获取书籍信息
getBookInfo:
{
    arguments:"document", // 传入的参数, 是从evaluate那边获取的页面的document 对象
    body:
    /**
     * 修改下面的代码, 以适配你的网站
     */

    // 此处是函数体, 这里存放爬虫代码, 使用 document 以及css 选择器来获取数据
    // 获取并透过 return 返回以下数据: {bookname, img, author, intro}
    let bookname = document.querySelector('.book-info h1 em').textContent; // 获取小说名字
    let img = document.querySelector('.book-img img').src; // 获取小说封面
    let author = document.querySelector('.book-info h1 .writer').textContent; // 获取小说作者
    let intro = document.querySelector('.intro').textContent; // 获取小说简介
    return {bookname, img, author, intro}
}
```

同理，在小说章节页面获取章节信息的dom，在json里写入配置函数

```
// 从小说正文获取章节内容
getContent:
{
    arguments:"document", // 传入的参数, 是从evaluate那边获取的页面的document 对象
    body:
    /**
     * 修改下面的代码, 以适配你的网站
     */

    let content = document.querySelectorAll('.content .content-text').map(element => element.innerText.trim()).join('')
    let title = document.querySelector('.content-text').innerText.replace(/\n|\r/g, '');
    let nextPageUrl = null;

    try {
        content = Array.from(document.querySelectorAll('.content .content-text')).map(element => element.innerText.trim()).join('')
    } catch(e){content = null}

    try {
        title = document.querySelector('.content-text').innerText.replace(/\n|\r/g, '');
    } catch(e){title = null}

    try {
        nextPageUrl = document.querySelector('div.nav-btn-group :last-child').href; // 获取下一页的链接
    } catch(e){nextPageUrl = null}

    return {content, title, nextPageUrl}
}

// 测试函数, 用于测试
"test":
```

## 循环章节/存储小说

在爬取完当前章节内容后，要构造出下一章的内容。这里针对于不同网站有两种方式去完成

- 针对于下一章按钮是a标签的，获取到a标签的href，直接跳转这个页面即可。代表网站：起点、纵横、晋江、笔趣阁、noveltoon、wattpad等

```
a[href="https://www.xbiquge.tw/book/45525/40007998.html"]#link-next
```

```
// 应用 react flutter 前端 项目地址 打包发布 jsb 音乐/影视 ProcessOn - 免费... Alchitos Discover - Kibana RegEx: Learn, Build, Test puppeteer 无头模...
```

```
元素 控制台 源代码 网络 性能 内存 应用 Lighthouse 图层 Redux
```

```
<!DOCTYPE html>
<html>
  <head> ...
  </head>
  <body>
    <div id="wrapper" style="height: auto !important;">
      <div class="content_read" style="height: auto !important;">
        <div id="box_con" style="height: auto !important;">
          <div class="con_top"> ...
          <div class="bookname"> ...
          <div id="tlist" style="padding: 10px 0px; border-top: 1px dashed #e5e5e5; margin-top: 5px;"> ...
          <center> ...
            <div id="content" name="content"> ...
            <center class="clear"> ...
            <div class="bottom">
              <a href="javascipt:;" onclick="vote('45525');">推荐本书</a>
              <a href="https://www.xbiquge.tw/book/45525/40007996.html" id="link-preview">上一章</a>
              <a href="https://www.xbiquge.tw/book/45525/" id="link-index" title="神印王座II皓月当空章节目录>章节目录"> ...
              <a href="https://www.xbiquge.tw/book/45525/40007998.html" id="link-next">下一章</a>
            
```

```
1 // api  
2 await page.goto(url)
```

- 针对于下一章按钮不是a标签，而是通过click事件跳转的，可以模拟浏览器的点击事件进行跳转。  
代表网站：**番茄小说网**

The screenshot shows the developer tools of a web browser displaying the source code for a chapter page. The code includes a script that handles the click event for the 'next' button. The button is identified by the class 'muye-reader-btns .next'. The click event triggers a function that sets the href attribute of the button to the URL of the next chapter and then performs a page reload.

```
<html lang="zh">  
<head>...</head>  
<body>  
  <noscript>You need to enable JavaScript to run this app.</noscript>  
  <div id="app">  
    <div class="muye-reader">  
      <div class="muye-reader-inner">  
        <div class="muye-reader-exit"></div>  
        <div id="muye-reader-catalog-wrap">...</div>  
        <a href="https://www.changdunovel.com/" class="muye-reader-operation phone loop">...</a>  
        <div class="muye-reader-top"></div>  
        <div class="muye-reader-box font-NNMRHsV173Pd4pgy" style="font-family: DNMRHsV173Pd4pgy, -apple-system, "PingFang SC", "Microsoft YaHei", "Helvetica Neue", Helvetica, Arial, sans-serif;">  
          <h1 class="muye-reader-title">第2章 病榻前,得爷爷临终赠宝</h1>  
          <div class="muye-reader-desc">...</div>  
          <div class="muye-reader-content noselect">...</div>  
          <div class="muye-reader-btns">  
            <button class="byte-btn byte-btn-dashed byte-btn-size-large byte-btn-shape-square.muye-button.chapter-btn last" type="button">...</button>  
            <button class="byte-btn byte-btn-primary byte-btn-size-large byte-btn-shape-square.muye-button.chapter-btn.next" type="button">...</button>  
            <span>下一章</span>  
          </div>  
        </div>  
        </div>  
      </div>  
    </div>  
  </div>  
<script nonce="..."></script>  
<script nonce="text/javascript" src="//lf-cdn-tos.bytescm.com/obj/static/toutiao/muye/Reader_971f5a33.js"></script>  
<script nonce="..."></script>  
<script src="//lf-cdn-tos.bytescm.com/obj/static/toutiao/muye/js/index_cb0edf63.js"></script>  
</div>  
<div>...</div>
```

```
1 // api  
2 let a = document.querySelector(".muye-reader-btns .next");  
3 a.click();
```

## 下面是一个小说爬虫流程示例

```
index.js -- novel-crawler
index.js > ...
5  const {
6    cookies,
7    sleep,
8    sanitizeCookie,
9    findBookSourceByHost,
10   writeFile,
11   sanitizeFileName,
12 } = require("./utils");
13
14 const VERSION = "1.0.0";
15 // puppeteer config
16 const config = (function (filePath) {
17   let configFile = null;
18   if (!fs.existsSync(filePath)) {
19     console.log(`> 配置文件不存在，创建默认配置文件...`);
20     configFile = fs.readFileSync("./config.hjson.template", "utf8");
21     fs.writeFileSync(filePath, configFile, "utf8");
22   } else configFile = fs.readFileSync(filePath, "utf8");
23
24   return Hjson.parse(configFile);
25 })("./config.hjson");
26
```

The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit <https://support.apple.com/kb/HT200050>.

## 接入字体混淆如何爬取

番茄小说网接入了字体混淆，返回的章节内容很多都是特殊字符。我们再直接拿取dom结构就沒用了，可以用两种方式解决

- 硬碰硬，解密字体混淆方法。分析字体文件破译字体加密方案。
- ocr识别，puppeteer截图后，ocr识别文本

第2章 病榻前，得爷爷临终赠宝

书名：凡骨 作者：夏更大师 本章字数：2273字 更新时间：2022-04-02 11:21:32

“听过的。”

太平点了点头，随后一脸神往地说道：

“村口的赵叔说，离我们村不远的云庐山上就有仙人。前些年，清水镇的那头蛇妖，就是山上的仙人斩杀了，飞天遁地，好不威风。”

老人闻言呵呵一笑，随即问道：

“那太平你想跟那些仙人一样，长生不老，飞天遁地吗？”

“想啊，当然想。”

小太平毫不犹豫地点了点头，不过马上又不好意思地挠头笑道：

“不过赵叔说了，想要修行，须得天生灵骨才行。若你是天生灵骨，你的名字便会自动出现在仙籍之列，到时候会有仙人，亲自下山来接你上山修行，否则一介凡骨，是修不了仙的。”

听得出来，太平的语气其实有些失落。

“太平啊……”

老人犹豫了一下，随后一脸认真地看向太平，压低声音问道：

“若是有一段机缘，能让你得到仙籍，但须得冒着丧命的风险，你可否愿意？”

“能让我……得到仙籍？”

小太平先是一脸愕然，继而开始认真地思忖了起来。

... (remaining content of the chapter)

在写番茄小说网的爬虫用到的是ocr识别方案。可以使用本地的ocr模块或者直接调用各大云的ocr接口，识别准确率都接近100%（因为小说纯文字识别，没啥难度）。

下面是爬取番茄的示例。

```

fanqie.js ×
fanqie.js > @<function> > @page.evaluate() callback
13 // 调试时前往控制台 https://console.cloud.tencent.com/cam/api 进行观察
14 const clientConfig = {
15   credential: {
16     secretId: "AKIDgnv7xb7RbcS7f3GPV9duxoESTSGt0hw",
17     secretKey: "WXX05Y5RchdZNg5wpvseCakiT2nNt3zd",
18   },
19   region: "ap-beijing",
20   profile: {
21     httpProfile: {
22       endpoint: "ocr.tencentcloudapi.com",
23     },
24   },
25 };
26
27 // 实例化要调用的产品的client对象,clientProfile是可选的
28 const client = new OcrClient(clientConfig);
29
30 const ocrText = async () => {
31   const image = fs.readFileSync("./example.jpg").toString("base64");
32
33   const result = await client.recognizeText({
34     Image: {
35       Base64: image,
36     },
37     SubType: "COMMON_TYPE",
38   });
39
40   const textList = result.TextBlockList;
41
42   console.log(`Detected ${textList.length} text blocks`);
43
44   for (const textBlock of textList) {
45     const text = textBlock.Text;
46     const location = textBlock.Location;
47
48     console.log(`Text: ${text}`);
49     console.log(`Location: ${location}`);
50   }
51
52   return textList;
53 }
54
55 ocrText().then(result => {
56   console.log(result);
57 })
58
59 module.exports = {
60   fanqie: fanqie,
61 };

```

## 五、反爬虫和反反爬虫策略

没有100%防范爬虫的方案，反爬虫的主要目的是增加爬虫的成本，尽可能促使其放弃爬取。

下面介绍下本文写作中遇到的一些反爬虫手段以及如何去解决。

## 真实请求检测

- 反爬策略：**头条的feed流接口会判断请求是否真实。所以我们应当使爬虫请求尽可能地模拟用户真实访问。
- 反反爬：**可以采取下面的方法
  - 携带大量请求头，并随机变化。可以本地写一个request header的数组，每次请求随机带上一些，或者使用第三方fake header库也可以
  - 携带页面的cookie，模拟登录态
  - 用无头浏览器爬取，例如js的puppeteer，python的selenium

## 请求间隔检测

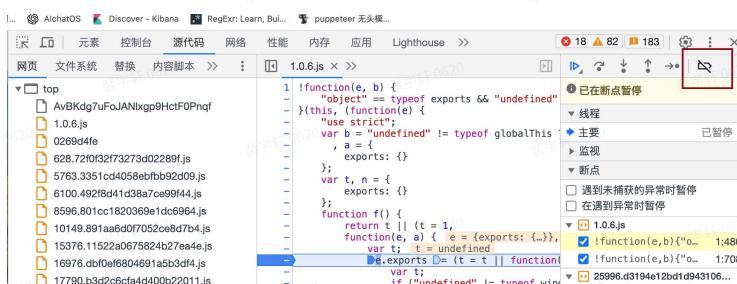
- 反爬策略：**一般接口都会将时间间隔固定的请求判定为爬虫。
- 反反爬：**我们可以每次发送请求后，sleep随机等待时间

## js加密

- 反爬策略：**头条feed请求会携带as、cp两个加密参数。
- 反反爬：**具体计算过程在页面源码中存在，当然源码经过了js混淆。于是可以debug找到加密部分的代码，并破解它。这也是js逆向的过程。

## 控制台无限debugger

- 反爬策略：**晋江的页面打开devtools会自动debugger卡住。
- 反反爬：**解决方法
  - 我们可以关掉devtools的开启debugger功能，这样就可以看页面的源码了



```
1 // UNINTIALIZED MODULE: ./react-devtools-shim/strictMode.js
2 // Install the hook on window, which is an event emitter.
3 * Note: this global hook, REACT_DEVTOOLS_GLOBAL_HOOK_, is a dependency.
4 * It's especially important to avoid creating direct dependency.
5 * That's why we still inline the whole event emitter implementation.
6 * the string format implementation, and part of the console implementation.
7 */
8
9 function installHook(target) {
10   if (!target || !target.hasOwnProperty('__REACT_DEVTOOLS_GLOBAL_HOOK__')) {
11     target = window;
12     target.__REACT_DEVTOOLS_GLOBAL_HOOK__ = {};
13   }
14   target.onerror = function(e) {
15     if (e.message === 'Uncaught ReferenceError: debugger is not defined') {
16       return;
17     }
18     if (e.message === 'Breakpoint hit') {
19       target.console.log('Breakpoint hit');
20     }
21   };
22   target.addEventListener('error', function(e) {
23     if (e.message === 'Uncaught ReferenceError: debugger is not defined') {
24       target.console.log('Breakpoint hit');
25     }
26   });
27   target.addEventListener('unhandledrejection', function(e) {
28     if (e.reason.message === 'Uncaught ReferenceError: debugger is not defined') {
29       target.console.log('Breakpoint hit');
30     }
31   });
32   target.addEventListener('message', function(e) {
33     if (e.data.type === 'break') {
34       target.console.log('Breakpoint hit');
35     }
36   });
37   target.addEventListener('devicemotion', function(e) {
38     if (e.message === 'Breakpoint hit') {
39       target.console.log('Breakpoint hit');
40     }
41   });
42   target.addEventListener('devicelight', function(e) {
43     if (e.message === 'Breakpoint hit') {
44       target.console.log('Breakpoint hit');
45     }
46   });
47   target.addEventListener('storage', function(e) {
48     if (e.message === 'Breakpoint hit') {
49       target.console.log('Breakpoint hit');
50     }
51   });
52   target.addEventListener('storageerror', function(e) {
53     if (e.message === 'Breakpoint hit') {
54       target.console.log('Breakpoint hit');
55     }
56   });
57   target.addEventListener('storageevent', function(e) {
58     if (e.message === 'Breakpoint hit') {
59       target.console.log('Breakpoint hit');
60     }
61   });
62   target.addEventListener('storageerror', function(e) {
63     if (e.message === 'Breakpoint hit') {
64       target.console.log('Breakpoint hit');
65     }
66   });
67   target.addEventListener('storageevent', function(e) {
68     if (e.message === 'Breakpoint hit') {
69       target.console.log('Breakpoint hit');
70     }
71   });
72   target.addEventListener('storageerror', function(e) {
73     if (e.message === 'Breakpoint hit') {
74       target.console.log('Breakpoint hit');
75     }
76   });
77   target.addEventListener('storageevent', function(e) {
78     if (e.message === 'Breakpoint hit') {
79       target.console.log('Breakpoint hit');
80     }
81   });
82   target.addEventListener('storageerror', function(e) {
83     if (e.message === 'Breakpoint hit') {
84       target.console.log('Breakpoint hit');
85     }
86   });
87   target.addEventListener('storageevent', function(e) {
88     if (e.message === 'Breakpoint hit') {
89       target.console.log('Breakpoint hit');
90     }
91   });
92   target.addEventListener('storageerror', function(e) {
93     if (e.message === 'Breakpoint hit') {
94       target.console.log('Breakpoint hit');
95     }
96   });
97   target.addEventListener('storageevent', function(e) {
98     if (e.message === 'Breakpoint hit') {
99       target.console.log('Breakpoint hit');
100    }
101   });
102   target.addEventListener('storageerror', function(e) {
103     if (e.message === 'Breakpoint hit') {
104       target.console.log('Breakpoint hit');
105     }
106   });
107   target.addEventListener('storageevent', function(e) {
108     if (e.message === 'Breakpoint hit') {
109       target.console.log('Breakpoint hit');
110     }
111   });
112   target.addEventListener('storageerror', function(e) {
113     if (e.message === 'Breakpoint hit') {
114       target.console.log('Breakpoint hit');
115     }
116   });
117 }
```

- Never pause here: 在debugger位置的行号处，右键Never pause here，永远不在debugger处运行
- 条件断点，设置条件断点为false也可以跳过debugger;

The screenshot shows a code editor with a sidebar containing file names and a main pane displaying a script. A blue arrow points to line 107 of the script. A tooltip over the line says "Line 107: 条件断点 ▾". Below the line, the word "false" is highlighted in red, indicating it is the condition for the breakpoint. The script content includes comments about avoiding direct dependencies and inline event emitter implementations.

```
100 * It's especially important to avoid creating direct dependency c
101 * That's why we still inline the whole event emitter implementati
102 * the string format implementation, and part of the console imple
103 *
104 *
105 */
106 function installHook(target) {
107 if (target.__hasOwnProperty('__REACT_DEVTOOLS_GLOBAL_HOOK__'))
108     return null;
109 }
110 let targetConsole = console;
111 let targetConsoleMethods = {};
```

## Never pause here

在 debugger 位置，点击行号，右键 Never pause here，永远不在此处断下即可：

## IP检测

- **反爬策略：**接口层会将短时间大量同一ip请求判定为爬虫。
- **反反爬：**我们可以构造ip池，每次请求随机使用一个。

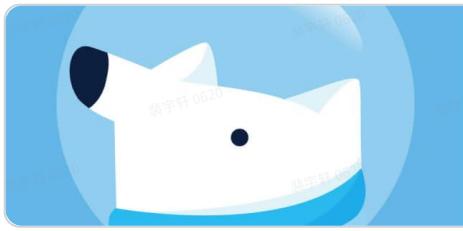
## 图形验证码

- **反爬策略：**一段时间后，页面可能会出现验证码识别，要完成识别后才能进行下一步操作。
- **反反爬：**使用ocr识别验证码，再进行后续操作。当然使用云厂商或本地搭建都可以。

## 字体混淆

- **反爬策略：**番茄小说内容就接入了字体混淆
- **反反爬：**使用ocr识别或者破解字体加密手段

下面也列举了知乎的一些回答，有兴趣可以看看。



知乎 <https://www.zhihu.com/question/288736516>

## 爬虫中遇到过哪些厉害的反爬，以及骚操作反反爬？ - 知乎

本人python菜鸟一枚，专业相关。平日里喜欢爬一些电影电视剧还有其他，突然想请教一下各位小哥哥大神，平…



知乎 <https://www.zhihu.com/question/58342241>

## 有哪些有趣的反爬虫手段？ - 知乎

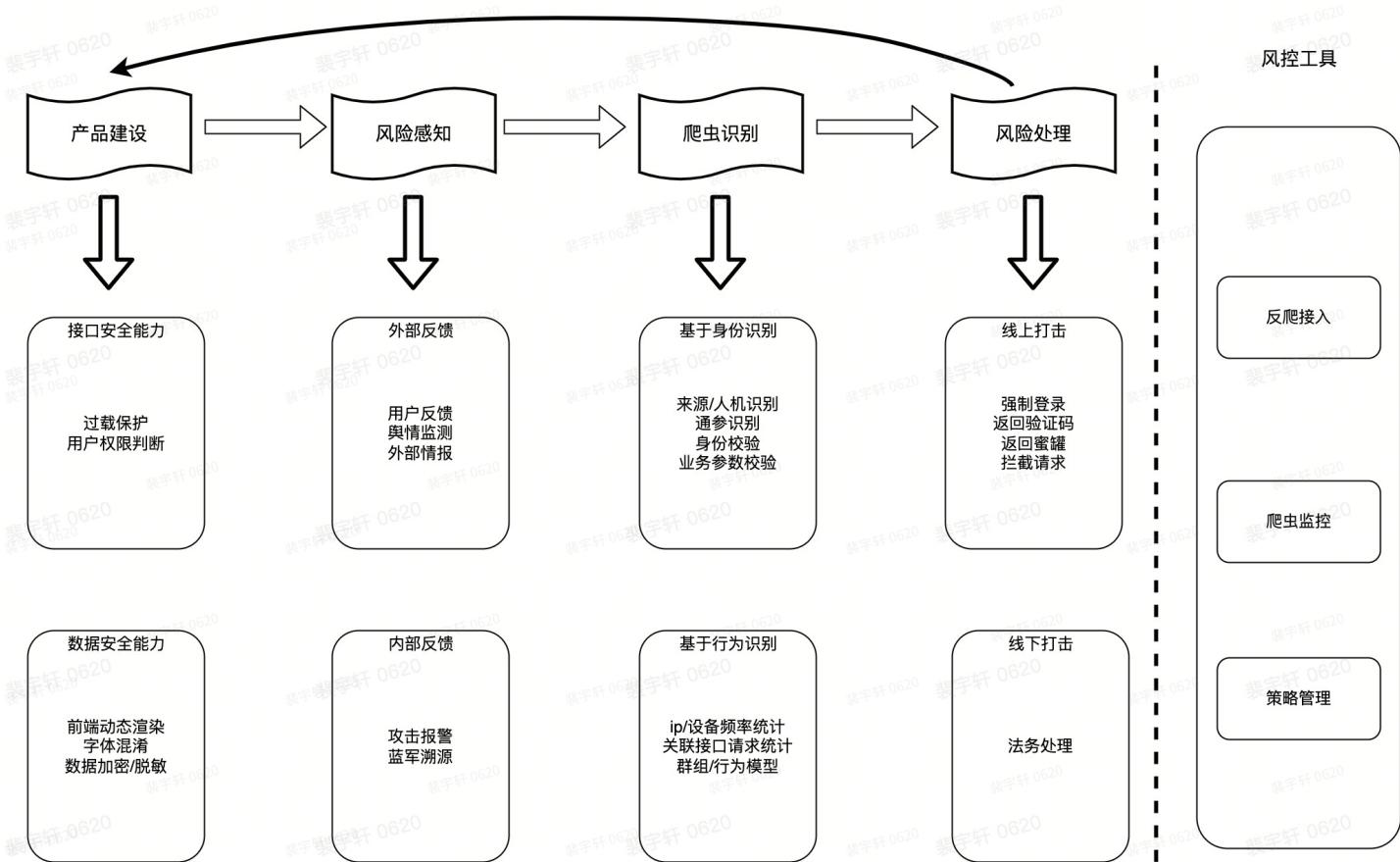
爬虫与反爬虫的斗争从未停止。反爬虫程序员一直在与爬虫程序员做斗争~那么，大家都见过什么有趣的反爬虫…

## 六、理想反爬系统架构

一个理想的反爬架构要从多个方面入手

- 产品建设
- 风险感知
- 爬虫识别
- 风险处理

### 理想的反爬系统框架



## 七、爬虫相关法律规定

关于爬虫的法律规定主要集中在以下几类法规中：

- 法律：  
  - 《刑法》扰乱公共秩序罪，第 285 条，违反国家规定，侵入国家事务、国防建设、尖端科学技术领域的计算机信息系统的，处三年以下有期徒刑或者拘役。违反国家规定，侵入前款规定以外的计算机信息系统或者采用其他技术手段，获取该计算机信息系统中存储、处理或者传输的数据，或者对该计算机信息系统实施非法控制，情节严重的，处三年以下有期徒刑或者拘役，并处或者单处罚金；情节特别严重的，处三年以上七年以下有期徒刑，并处罚金。
- 法律：一般法律
  - 《中华人民共和国个人信息保护法》
  - 《中华人民共和国网络安全法》
- 行政法规
  - 2019年《数据安全管理办办法（征求意见稿）》第十六条 网络运营者采取自动化手段访问收集网站数据，不得妨碍网站正常运行；此类行为严重影响网站运行，如自动化访问收集流量超过网站日均流量三分之一，网站要求停止自动化访问收集时，应当停止。

## 八、novel-crawler总结

- 一个node命令行爬虫项目，只需输入要爬取小说的网页地址，即可爬取整本小说
- 目前支持小说网站（包括了国内、国际化所有的竞品）
  - 国内
    - 起点
    - 晋江
    - 七猫
    - 纵横
    - 番茄
    - 笔趣阁
  - 国外
    - noveltoon
    - wattpad

- 如果你有感兴趣的文学网站，可以仿照bookSource下的json配置添加网站的爬取方式（相信对大家来说是小case），或者提个issue、私聊@我 裴宇轩 也可以，我有时间会将该网站的爬虫配置加进去。
- 项目中的toutiao.js是用来爬取头条feed流和大V主页的，从feed流接口拿到大V的uid，再去爬取大V个人主页的全部信息
- Gitlab地址：**<https://code/byted.org/novel-fe/novel-crawler>
- 最后再来一次免责说明：爬虫写的好，牢饭吃到饱，不要用此爬虫脚本来进行盈利或破坏行为！

### 免责声明

本文章涉及到的应用仅供学习交流使用，不得用于任何商业用途，数据来源于互联网公开内容，没有获取任何私有和有权限的信息（个人信息等）。由此引发的任何法律纠纷与本人无关！禁止将本文技术或者本文所关联的Gitlab项目源码用于任何目的。